

| | | |
|-----|-----|--------------|
| (1) | 51 | DECLARATIONS |
| (1) | 112 | R/W PSECT |
| (1) | 190 | SATSSS80 |
| (1) | 239 | PURGWS TESTS |
| (2) | 331 | REG_SAVE |
| (2) | 352 | REG_CHECK |
| (2) | 394 | PRINT_FAIL |
| (2) | 441 | MODE_ID |


```
0000 1 .TITLE SATSSS80, - SATS SYSTEM SERVICE TESTS (SUCC S.C.)
0000 2 .IDENT 'V04-000'
0000 3
0000 4
0000 5 *****
0000 6
0000 7 *
0000 8 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 9 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 10 * ALL RIGHTS RESERVED.
0000 11 *
0000 12 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 13 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 14 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 15 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 16 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 17 * TRANSFERRED.
0000 18 *
0000 19 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 20 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 21 * CORPORATION.
0000 22 *
0000 23 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 24 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 25 *
0000 26 *****
0000 27
0000 28
0000 29 :++
0000 30 : FACILITY: SATS SYSTEM SERVICE TESTS
0000 31
0000 32 : ABSTRACT: The SATSSS80 module tests the execution of the following
0000 33 : VMS system services:
0000 34 :
0000 35 : $PURGWS
0000 36 :
0000 37 : ENVIRONMENT: User mode image.
0000 38 : Needs CMKRNL privilege and dynamically acquires other
0000 39 : privileges, as needed.
0000 40 :
0000 41 : AUTHOR: Larry D. Jones, CREATION DATE: JULY, 1979
0000 42 :
0000 43 : MODIFIED BY:
0000 44 :
0000 45 : V03-002 KDM0002 Kathleen D. Morse 28-Jun-1982
0000 46 : Added $PRVDEF.
0000 47 :
0000 48 : **
0000 49 : --
```

```

0000 51      .SBTTL  DECLARATIONS
0000 52      :
0000 53      : MACRO LIBRARY CALLS
0000 54      :
0000 55      .LIBRARY /SYSS$LIBRARY:STARLET.MLB/
0000 56      $JPIDEF      : GETJPI definitions
0000 57      $PHDDEF      : process header definitions
0000 58      $PRVDEF      : Privilege bit definitions
0000 59      $SHR MESSAGES UETP,116,<<TEXT,INFO>> : UETPS TEXT definition
0000 60      $SFDEF       : stack frame definitions
0000 61      $SSDEF       : system status code definitions
0000 62      $STSDEF      : STS definitions
0000 63      $UETPDEF     : UETP message definitions
0000 64      :
0000 65      : Equated symbols
0000 66      :
00000000 0000 67 WARNING      = 0      : warning severity value for msgs
00000001 0000 68 SUCCESS      = 1      : success
00000002 0000 69 ERROR        = 2      : error
00000003 0000 70 INFO         = 3      : information
00000004 0000 71 SEVERE       = 4      : fatal
0000 72      :
0000 73      :
0000 74      : MACROS
0000 75      :

```

```
00000000 77 .PSECT RODATA,RD,NOWRT,NOEXE,PAGE
0000 78
0000 79 TEST_MOD_NAME:
30 38 53 53 53 54 41 53 00' 0000 80 .ASCIC /SATSSS80/ ; needed for SATSMS message
08 0000
0009 81 TEST_MOD_NAME D:
53 53 53 54 41 53 00000011'010E0000' 0009 82 .ASCID /SATSSS80/ ; module name
30 38 0017
0019 83 TEST_MOD_BEGIN: ; start end and fail messages
6E 75 67 65 62 00' 0019 84 .ASCIC /begun/
05 0019
001F 85 TEST_MOD_SUCC:
6C 75 66 73 73 65 63 63 75 73 00' 001F 86 .ASCIC /successful/
0A 001F
002A 87 TEST_MOD_FAIL:
64 65 6C 69 61 66 00' 002A 88 .ASCIC /failed/
06 002A
0031 89 CS1: ; failure messages
21 20 74 73 65 54 00000039'010E0000' 0031 90 .ASCID \Test !AC service name !AC step !UL failed.\
6E 20 65 63 69 76 72 65 73 20 43 41 003F
70 65 74 73 20 43 41 21 20 65 6D 61 004B
2E 64 65 6C 69 61 66 20 4C 55 21 20 0057
0063 91 CS2:
74 63 65 70 78 45 0000006B'010E0000' 0063 92 .ASCID \Expected !AS = !XL received !AS = !XL\
4C 58 21 20 3D 20 53 41 21 20 64 65 0071
41 21 20 64 65 76 69 65 63 65 72 20 007D
4C 58 21 20 3D 20 53 0089
0090 93 CS3:
74 63 65 70 78 45 00000098'010E0000' 0090 94 .ASCID \Expected !AS!UB = !XL received !AS!UB = !XL\
20 3D 20 42 55 21 53 41 21 20 64 65 009E
64 65 76 69 65 63 65 72 20 4C 58 21 00AA
58 21 20 3D 20 42 55 21 53 41 21 20 00B6
4C 00C2
00C3 95 CS5:
69 20 65 64 6F 4D 000000CB'010E0000' 00C3 96 .ASCID \Mode is !AS.\
2E 53 41 21 20 73 00D1
00D7 97 EXP:
73 75 74 61 74 73 000000DF'010E0000' 00D7 98 .ASCID \status\
00E5 99 UM: ; mode messages
72 65 73 75 000000ED'010E0000' 00E5 100 .ASCID \user\
00F1 101 MSGVEC:
00000003 00F1 102 .LONG 3 ; PUTMSG message vector
00741133 00F5 103 .LONG UETPS_TEXT
00000001 00F9 104 .LONG 1
0000016F' 00FD 105 .ADDRESS MESSAGEL
0101 106 PURGWS:
53 57 47 52 55 50 00' 0101 107 .ASCIC /PURGWS/ ; service name
06 0101
0108 108 WS_STR:
70 20 63 6F 72 70 00000110'010E0000' 0108 109 .ASCID /proc pg cnt/
74 6E 63 20 67 0116
```



```
011B 111 ;
011B 112 ;.SBTTL R/W PSECT
00000000 113 ;.PSECT RWDATA,RD,WRT,NOEXE,PAGE
0000 114 ;
0000 115 ;PID:
00000000 0000 116 .LONG 0 ; PID for this process
0004 117 CURRENT_TC: ; ptr to current test case
00000000 0004 118 .LONG 0 ; put it on a long word boundry
0008 119 .ALIGN LONG
00000044 0008 120 REG_SAVE_AREA:
0008 121 .BLKL 15 ; register save area
007480D9 0044 122 MOD_MSG_CODE:
0048 123 .LONG UETPS_SATSMS ; test module message code for putmsg
00000000' 0048 124 TMN_ADDR:
004C 125 .ADDRESS TEST_MOD_NAME
00000019' 004C 126 TMD_ADDR:
0050 127 .ADDRESS TEST_MOD_BEGIN
00 0050 128 PRVPRT:
0051 129 .BYTE 0 ; protection return byte for SETPRT
00000000 00000000 0051 130 PRIVMASK:
0059 131 .QUAD 0 ; priv. mask
00000000 0059 132 CHM_CONT:
005D 133 .LONG 0 ; change mode continue address
00000065 005D 134 RETADR:
0065 135 .BLKL 2 ; returned address's from SETPRT
00000000 0065 136 STATUS:
0069 137 .LONG 0
00000000 0069 138 MODE:
006D 139 .LONG 0 ; current mode string pointer
74 73 69 67 65 72 00000075'010E0000' 006D 140 REG:
52 20 72 65 007B 141 .ASCID \register R\
007F 142 REGNUM:
00000000 007F 143 .LONG 0 ; register number
0083 144 MSGL:
00000050 0083 145 .LONG 80 ; buffer desc.
0000008B' 0087 146 .ADDRESS BUF
008B 147 BUF:
000000DB 008B 148 .BLKB 80
00DB 149 ML:
00000000 00DB 150 .LONG 0 ; desc. for BUF_CHECK routine
000000EB' 00DF 151 .ADDRESS GETBUF+8
00E3 152 GETBUF:
00000084 00E3 153 .LONG 132
000000EB' 00E7 154 .ADDRESS +4
0000016F 00EB 155 .BLKB 132
016F 156 MESSAGEL:
00000000 016F 157 .LONG 0 ; message desc.
0000008B' 0173 158 .ADDRESS BUF
0177 159 SERV_NAME:
00000000 0177 160 .LONG 0 ; service name pointer
017B 161 MSGVEC1: ; PUTMSG message vector
00000003 017B 162 .LONG 3
00741133 017F 163 .LONG UETPS_TEXT
00000001 0183 164 .LONG 1
00000000 0187 165 .LONG 0
018B 166 GET_LIST:
```

```
0004 018B 167 .WORD 4 ; GETJPI item list
030D 018D 168 .WORD JPIS_PPGCNT
0000019B 018F 169 .LONG PPG_CNT
00000000 0193 170 .LONG 0
00000000 0197 171 .LONG 0
00000000 019B 172 PPG_CNT: .LONG 0 ; before WS peak
00000000 019B 173 PPG_CNT1: .LONG 0 ; after WS peak
00000000 019F 174 .LONG 0
00000000 01A3 176 PURGE_AREA:
00000000 01A3 177 .ADDRESS TOUCH_PAGE ; PURGWS address block
00000000 01A7 178 .ADDRESS TOUCH_PAGE
00000000 01AB 179 LOCK_AREA:
00000000 01AB 180 .ADDRESS TEST_MOD_NAME ; LCKPAG address array
000003C6 01AF 181 .ADDRESS TEST_END
00000000 01B3 182 PURG:
00000000 01B3 183 $PURGWS PURGE_AREA ; PURGWS parameter list
00000000 184 .PSECT TOUCH_PAGE, RD, PAGE
0000 185 .ALIGN PAGE
0000 186 TOUCH_PAGE:
00000600 0000 187 .BLKB 1536 ; 3 pages to touch
```



```
00000000 189      .PSECT SATSSS80, RD, WRT, EXE, PAGE
0000      190      .SBTTL SATSSS80
0000      191      :++
0000      192      : FUNCTIONAL DESCRIPTION:
0000      193      :
0000      194      :     After performing some initial housekeeping, such as
0000      195      :     printing the module begin message and acquiring needed privileges,
0000      196      :     the system services are tested in each of their normal conditions.
0000      197      :     Detected failures are identified and an error message is printed
0000      198      :     on the terminal. Upon completion of the test a success or fail
0000      199      :     message is printed on the terminal.
0000      200      :
0000      201      : CALLING SEQUENCE:
0000      202      :
0000      203      :     $ RUN SATSSS80 ... (DCL COMMAND)
0000      204      :
0000      205      : INPUT PARAMETERS:
0000      206      :
0000      207      :     none
0000      208      :
0000      209      : IMPLICIT INPUTS:
0000      210      :
0000      211      :     none
0000      212      :
0000      213      : OUTPUT PARAMETERS:
0000      214      :
0000      215      :     none
0000      216      :
0000      217      : IMPLICIT OUTPUTS:
0000      218      :
0000      219      :     Messages to SYS$OUTPUT are the only output from SATSSS80.
0000      220      :     They are of the form:
0000      221      :
0000      222      :     XUETP-S-SATSMS, TEST MODULE SATSSS80 BEGUN ... (BEGIN MSG)
0000      223      :     XUETP-S-SATSMS, TEST MODULE SATSSS80 SUCCESSFUL ... (END MSG)
0000      224      :     XUETP-E-SATSMS, TEST MODULE SATSSS80 FAILED ... (END MSG)
0000      225      :     XUETP-I-TEXT, ... (VARIABLE INFORMATION ABOUT A TEST MODULE FAILURE)
0000      226      :
0000      227      : COMPLETION CODES:
0000      228      :
0000      229      :     The SATSSS80 routine terminates with a $EXIT to the
0000      230      :     operating system with a status code defined by UETP$_SATSMS.
0000      231      :
0000      232      : SIDE EFFECTS:
0000      233      :
0000      234      :     none
0000      235      :
0000      236      : --
0000      237      :
0000      238      : TEST_START SATSSS80                                : let the test begin
```

```
0000 0000
0004'CF 00 DD 0002
0000'CF 00 DF 0006
00000000'GF 02 FB 000C
00000000'GF 00 FB 0013
0009'CF 7F 001A
00000000'GF 01 FB 001E
037A 30 0025
004C'CF 001F'CF DE 0028
0044'CF 03 00 01 FO 002F
00 DD 0036
0265'CF 01 FB 0038
003D 239 STP0:
003D 240 .SBTTL PURGWS TESTS
003D 241 :+
003D 242 $PURGWS tests
003D 243 :
003D 244 test _S form with a dry WS and adr array elements =
003D 245 :-
003D 246
0177'CF 0101'CF DE 003D 247 MOVAL W^PURGWS,W^SERV_NAME ; set service name
0069'CF 00E5'CF DE 0044 248 MOVAL W^UM,W^MODE ; set the mode
004B 249 MODE TO,10$,KRNL,NOREGS ; get to kernel mode
59 00000000'9F DO 0068 250 MOVL @#CTL$GL_PHD,R9 ; get the process header adr
0051'CF 69 DE 006F 251 MOVAL PHD$Q PRIVMSK(R9),W^PRIVMASK ; get the priv. mask
0074 252 MODE FROM,T0$ ; return to user mode
0075 253 PRIV ADD,PSWAPM ; allow page locking
0095 254 PUSHL #0 ; push a dummy parameter
0265'CF 01 FB 0097 255 CALLS #1,W^REG_SAVE ; save a reg snapshot
009C 256 $LCKPAG_S INADR =W^LOCK_AREA ; nail down everything but TOUCH_PAG
00AB 257 $PURGWS_S INADR =W^PURGE_AREA ; squeeze the juice out of this proc
00B6 258 FAIL_CHECK $$$_NORMAL ; check for success
00B6 259 PUSHL #$$$_NORMAL
026F'CF 01 FB 00B8 260 CALLS #1,W^REG_CHECK
00BD 261 $GETJPI_S ITMLST=W^GET_LIST ; get the process page count in ques
00D2 262 :+
00D2 263 test _S form with adr array elements one page apart
00D2 264 :-
00D2 265 NEXT_TEST
00D2
0004'CF 01 DO 00D2 STP1:
00 DD 00D7 MOVL #1,W^CURRENT_TC
0265'CF 01 FB 00D9 PUSHL #0
01A7'CF 000001FF 8F CO 00DE 266 CALLS #1,W^REG_SAVE
018F'CF 019F'CF DE 00E7 267 ADDL2 #511,W^PURGE_AREA+4 ; set new adr array element
00EE 268 MOVAL W^PPG_CNT1,W^GET_LIST+4 ; point to a new storage location
00F9 269 $PURGWS_S INADR =W^PURGE_AREA ; squeeze blood out of a turnip
00F9 269 FAIL_CHECK $$$_NORMAL ; check for success
026F'CF 01 DD 00F9 PUSHL #$$$_NORMAL
019F'CF 019B'CF D1 0115 270 CALLS #1,W^REG_CHECK
271 $GETJPI_S ITMLST=W^GET_LIST ; get the new process page count
CMPL -W^PPG_CNT,W^PPG_CNT1 ; are they the same?
```



```

      019F'CF 11 13 011C 272      BEQL 10$      : br if they are
      019B'CF DD 011E 273      PUSHL W^PPG_CNT1 : push received
      0108'CF DD 0122 274      PUSHL W^PPG_CNT : push expected
      02B1'CF 03 DF 0126 275      PUSHAL W^WS_STR : push string variable
      FB 012A 276      CALLS #3,W^PRINT_FAIL : print the failure
      012F 277 10$:
      012F 278 :+
      012F 279 :
      012F 280 : test _G form with one page of juice in the process page count
      012F 281 :
      012F 282 :-
      012F 283      NEXT_TEST

      0004'CF 02 DD 012F      STP2:
      00  DD 0134      MOVL #2,W^CURRENT_TC
      0265'CF 01 FB 0136      PUSHL #0
      018F'CF 019B'CF DE 013B 284      MOVAL W^PPG_CNT,W^GET_LIST+4 : reset the process page pointer
      0000'CF D5 0142 285      TSTL W^TOUCH_PAGE : suck in a new page
      0146 286      $GETJPI_S ITMLST=W^GET_LIST : get page count after touch
      015B 287      $PURGWS-G W^PURG : try _G form
      0164 288      FAIL_CHECK $$$_NORMAL : check success
      0164 DD 0164      PUSHL #$$$_NORMAL
      026F'CF 01 FB 0166      CALLS #1,W^REG_CHECK
      018F'CF 019F'CF DE 016B 289      MOVAL W^PPG_CNT1,W^GET_LIST+4 : set new page count pointer
      0172 290      $GETJPI_S ITMLST=W^GET_LIST : get the new process page count
      0187 291      DECL W^PPG_CNT : create expected
      019B'CF 019B'CF D1 018B 292      CMPL W^PPG_CNT,W^PPG_CNT1 : did we squeeze a page out?
      019F'CF 11 13 0192 293      BEQL 20$      : br if yes
      019F'CF DD 0194 294      PUSHL W^PPG_CNT1 : push recieved
      019B'CF DD 0198 295      PUSHL W^PPG_CNT : push expected
      0108'CF DF 019C 296      PUSHAL W^WS_STR : push string variable
      02B1'CF 03 FB 01A0 297      CALLS #3,W^PRINT_FAIL : print the failure
      01A5 298 20$:
      01A5 299 :+
      01A5 300 :
      01A5 301 : test _S form with more than one page to recover
      01A5 302 :
      01A5 303 :-
      01A5 304      NEXT_TEST

      0004'CF 03 DD 01A5      STP3:
      00  DD 01AA      MOVL #3,W^CURRENT_TC
      0265'CF 01 FB 01AC      PUSHL #0
      01A7'CF 00000400 8F C0 01B1 305      ADDL2 #1024,W^PURGE_AREA+4 : make a three page purge area
      018F'CF 019F'CF DE 01BA 306      MOVAL W^PPG_CNT1,W^GET_LIST+4 : reset the process page pointer
      56 0000'CF DE 01C1 307      MOVAL W^TOUCH_PAGE,R6 : set a page pointer
      57 03 DD 01C6 308      MOVL #3,R7 : set a page count
      01C9 309 30$:
      01C9 310      TSTL (R6) : touch a page
      56 00000200 8F C0 01CB 311      ADDL2 #512,R6 : point to next page
      F4 57 FS 01D2 312      SOBGTR R7,30$ : do all pages
      00 DD 01D5 313      PUSHL #0 : push a dummy paramter
      0265'CF 01 FB 01D7 314      CALLS #1,W^REG_SAVE : save a reg snapshot
      01DC 315      $GETJPI_S ITMLST=W^GET_LIST : get the process page count
      01F1 316      $PURGWS_S INADR=W^PURGE_AREA : clean it up
```



```
01FC 317
01FC 317
01FE 318
0203 319
020A 319
021F 320
0224 321
022B 322
022D 323
0231 324
0235 325
0239 326
023E 327
023E 328
40$:
004C'CF DD 023E
0048'CF DD 0242
02 DD 0246
0044'CF DD 0248
00000000'GF 04 FB 024C
0044'CF 01 1C 01 FO 0253
0044'CF DD 025A
00000000'GF 01 FB 025E

FAIL_CHECK SSS_NORMAL ; check for success
PUSHL #SSS_NORMAL
CALLS #1,W*REG_CHECK
MOVAL W*PPG_CNT,W*GET_LIST+4 ; set new PPG pointer
$GETJPI_S ITM[ST=W*GET_LIST ; get new process page count
SUBL2 #3,W*PPG_CNT1 ; set expected PPGCNT
CMPL W*PPG_CNT,W*PPG_CNT1 ; did we get at least 3 pages?
BEQL 40$ ; br if OK
PUSHL W*PPG_CNT ; push recieved
PUSHL W*PPG_CNT1 ; push expected
PUSHAL W*WS_STR ; push string variable
CALLS #3,W*PRINT_FAIL ; print the failure

TEST_END
PUSHL W*TMD_ADDR
PUSHL W*TMN_ADDR
PUSHL #2
PUSHL W*MOD_MSG_CODE
CALLS #SST1,G*LIBSSIGNAL
INSV #1,#STSSV_INHIB_MSG,#1,W*MOD_MSG_CODE
PUSHL W*MOD_MSG_CODE
CALLS #1,G*SYS$EXIT
```

```
0265 331 .SBTTL REG_SAVE
0265 332
0265 333 :++
0265 334 : FUNCTIONAL DESCRIPTION:
0265 335 : Subroutine to save R2-R11 in the register save location.
0265 336
0265 337 : CALLING SEQUENCE:
0265 338 : PUSHL #0 ; save a dummy parameter
0265 339 : CALLS #1,W^REG_SAVE ; save R2-R11
0265 340
0265 341 : INPUT PARAMETERS:
0265 342 : NONE
0265 343
0265 344 : OUTPUT PARAMETERS:
0265 345 : NONE
0265 346 :--
0265 347
0265 348 REG_SAVE:
0265 349 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
0008'CF 14 AD 28 OFFC 0267 350 MOV C3 #4*10,^X14(FP),W^REG_SAVE_AREA ; save the registers in the program
026E 351 RET
026F 352 .SBTTL REG_CHECK
026F 353 :++
026F 354 : FUNCTIONAL DESCRIPTION:
026F 355 : Subroutine to test R0 & R2-R11 for proper content after a service
026F 356 : execution. A snapshot is taken by the REG_SAVE routine at the
026F 357 : beginning of each step and this routine is executed after the
026F 358 : services have been executed.
026F 359
026F 360 : CALLING SEQUENCE:
026F 361 : PUSHL #SS$ XXXXXX ; push expected R0 contents
026F 362 : CALLS #1,W^REG_CHECK ; execute this routine
026F 363
026F 364 : INPUT PARAMETERS:
026F 365 : expected R0 contents on the stack
026F 366
026F 367 : OUTPUT PARAMETERS:
026F 368 : possible error messages printed using $PUTMSG
026F 369 :--
026F 370
026F 371
026F 372 REG_CHECK:
026F 373 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
026F 374 CMPL 4(AP),R0 ; is this the right fail code?
0271 375 BEQL 10$ ; br if yes
0275 376 PUSHL R0 ; push received data
0277 377 PUSHL 4(AP) ; push expected data
0279 378 PUSHL W^EXP ; push the string variable
027C 379 CALLS #3,W^PRINT_FAIL ; print the error message
0280 380
0285 381 10$: CMPC3 #4*10,^X14(FP),W^REG_SAVE_AREA ; check all but R0
0285 382 BEQL 20$ ; br if O.K.
028C 383 SUBL3 #REG_SAVE_AREA,R3,R6 ; calculate the register number
028E 384 DIVL2 #4,R6
0296 385 ADDB3 #^X2,R6,-(SP) ; set number past R0-R1 and save
0299 386 BICL2 #3,R1 ; backup to register boundrys
029D 387 BICL2 #3,R3
```

50 04 AC D1 0271 374
OE 13 0275 375
50 DD 0277 376
04 AC DD 0279 377
00D7'CF DF 027C 378
02B1'CF 03 FB 0280 379

0008'CF 14 AD 28 29 0285 381
22 13 028C 382
56 53 00000008'BF C3 028E 383
56 04 C6 0296 384
7E 56 02 81 0299 385
51 03 CA 029D 386
53 03 CA 02A0 387

```

61 DD 02A3 388      PUSHL (R1)      ; push received data
63 DD 02A5 389      PUSHL (R3)      ; push expected data
006D'CF DF 02A7 390      PUSHAL W^REG ; set string pntr param.
02B1'CF 04 FB 02AB 391      CALLS #4,W^PRINT_FAIL ; print the error message
                                20$:
                                04 RET
                                .SBTTL PRINT_FAIL
                                :++
                                : FUNCTIONAL DESCRIPTION:
                                : Subroutine to report failures using $PUTMSG
                                :
                                : CALLING SEQUENCE:
                                : Mode #1      PUSHL EXPECTED Mode #2      PUSHL REG NUMBER
                                :              PUSHL RECEIVED              PUSHL EXPECTED
                                :              PUSHAL STRING VAR           PUSHL RECEIVED
                                :              CALLS #3,W^PRINT_FAIL       PUSHAL STRING VAR
                                :              CALLS #4,W^PRINT_FAIL
                                : Mode #3      PUSHAL STRING VAR
                                :              CALLS #1,W^PRINT_FAIL
                                :
                                : INPUT PARAMETERS:
                                : Listed above
                                :
                                : OUTPUT PARAMETERS:
                                : an error message is printed using $PUTMSG
                                :
                                : --
                                :
                                : PRINT_FAIL:
                                : WORD      ^M<R2,R3,R4,R5>
                                : $FAO_S   W^CS1,W^MESSAGEL,W^MSGL,#TEST_MOD_NAME,W^SERV_NAME,W^CURRENT_TC
                                : $PUTMSG_S W^MSGVEC ; print the message
                                : CMPB     (AP),#4 ; is this a register message?
                                : BEQL     10$ ; br if yes
                                : CMPB     (AP),#1 ; is this just a message?
                                : BEQL     20$ ; br if yes
                                : $FAO_S   W^CS2,W^MESSAGEL,W^MSGL,4(AP),8(AP),4(AP),12(AP)
                                : BRB      30$ ; goto output message
                                :
                                : 10$:
                                : $FAO_S   W^CS3,W^MESSAGEL,W^MSGL,4(AP),16(AP),8(AP),4(AP),16(AP),12(AP)
                                : BRB      30$ ; goto output message
                                :
                                : 20$:
                                : MOVL     4(AP),W^MSGVEC1+12 ; save string address
                                : $PUTMSG_S W^MSGVEC1 ; print the message
                                : BRB      40$ ; skip the other message
                                :
                                : 30$:
                                : $PUTMSG_S W^MSGVEC ; print the message
                                :
                                : 40$:
                                : CALLS    #0,W^MODE ID ; identify the mode
                                : MOVAL    W^TEST_MOD_FAIL,W^TMD_ADDR ; set failure message address
                                : INSV     #ERROR,#0,#3,W^MOD_MSG_CODE ; set severity code
                                : RET
003C 02B1 417      .WORD      ^M<R2,R3,R4,R5>
02B3 418      $FAO_S   W^CS1,W^MESSAGEL,W^MSGL,#TEST_MOD_NAME,W^SERV_NAME,W^CURRENT_TC
02D4 419      $PUTMSG_S W^MSGVEC ; print the message
04 6C 91 02E5 420      CMPB     (AP),#4 ; is this a register message?
26 13 02E8 421      BEQL     10$ ; br if yes
01 6C 91 02EA 422      CMPB     (AP),#1 ; is this just a message?
48 13 02ED 423      BEQL     20$ ; br if yes
40 11 02EF 424      $FAO_S   W^CS2,W^MESSAGEL,W^MSGL,4(AP),8(AP),4(AP),12(AP)
19 11 030E 425      BRB      30$ ; goto output message
0310 426 10$:
0310 427      $FAO_S   W^CS3,W^MESSAGEL,W^MSGL,4(AP),16(AP),8(AP),4(AP),16(AP),12(AP)
0335 428      BRB      30$ ; goto output message
0337 429 20$:
0187'CF 04 AC D0 0337 430      MOVL     4(AP),W^MSGVEC1+12 ; save string address
033D 431      $PUTMSG_S W^MSGVEC1 ; print the message
11 11 034E 432      BRB      40$ ; skip the other message
0350 433 30$:
0350 434      $PUTMSG_S W^MSGVEC ; print the message
0361 435 40$:
0044'CF 0375'CF 00 FB 0361 436      CALLS    #0,W^MODE ID ; identify the mode
004C'CF 002A'CF DE 0366 437      MOVAL    W^TEST_MOD_FAIL,W^TMD_ADDR ; set failure message address
0044'CF 03 00 02 FO 036D 438      INSV     #ERROR,#0,#3,W^MOD_MSG_CODE ; set severity code
04 0374 439      RET
```



```

0375 441 .SBTTL MODE_ID
0375 442 :++
0375 443 : FUNCTIONAL DESCRIPTION:
0375 444 : Subroutine to identify the mode that an exit handler is in.
0375 445 :
0375 446 : CALLING SEQUENCE:
0375 447 : CALLS #0,W^MODE_ID
0375 448 :
0375 449 : INPUT PARAMETERS:
0375 450 : MODE contains an address pointing to an ascii string desc.
0375 451 : of the current CPU mode.
0375 452 :
0375 453 : OUTPUT PARAMETERS:
0375 454 : NONE
0375 455 :
0375 456 :--
0375 457 :
003C 0375 458 MODE_ID:
0377 459 .WORD ^M<R2,R3,R4,R5>
0390 460 $FAO S W^CS5,W^MESSAGEL,W^MSGL,MODE ; format the error message
04 03A1 461 $PUTMSG_S W^MSGVEC ; print the mode message
RET

```

```
03A2 465 MOD_MSG_PRINT:
03A2 466 :
03A2 467 : *****
03A2 468 : *
03A2 469 : * PRINTS THE TEST MODULE BEGUN/SUCCESSFUL/FAILED MESSAGES *
03A2 470 : * (USING THE PUTMSG MACRO). *
03A2 471 : *
03A2 472 : *****
03A2 473 :
03A2 474 PUTMSG <MOD_MSG_CODE,#2,TMN_ADDR,TMD_ADDR> : PRINT MSG
05 03BD 475 RSB ; ...-AND RETURN TO CALLER
03BE 476 :
03BE 477 CHMRTN:
03BE 478 : *****
03BE 479 : *
03BE 480 : * CHANGE MODE ROUTINE. THIS ROUTINE GETS CONTROL WHENEVER *
03BE 481 : * A CMKRNL, CMEXEC, OR CMSUP SYSTEM SERVICE IS ISSUED *
03BE 482 : * BY THE MODE MACRO ('TO' OPTION). IT MERELY DOES *
03BE 483 : * A JUMP INDIRECT ON A FIELD SET UP BY MODE. IT HAS *
03BE 484 : * THE EFFECT OF RETURNING TO THE END OF THE MODE *
03BE 485 : * MACRO EXPANSION. *
03BE 486 : *
03BE 487 : *****
03BE 488 :
00000059'FF 0000 03BE 489 .WORD 0 ; ENTRY MASK
17 03C0 490 JMP @CHM_CONT ; RETURN TO MODE MACRO IN NEW MODE
03C6 491 :
03C6 492 : * RET INSTR WILL BE ISSUED IN EXPANSION OF 'MODE FROM, ....' MACRO
03C6 493 :
03C6 494 TEST_END:
03C6 495 .END SATSSS80
```

SATSSS80
Symbol table

J 7
- SATS SYSTEM SERVICE TESTS (SUCC S.C.) 16-SEP-1984 01:04:10 VAX/VMS Macro V04-00
5-SEP-1984 04:33:42 [UETPSY.SRC]SATSSS80.MAR;1

Page 14
(3)

| | | | |
|----------------|------------|----|----|
| \$\$ARGS | = 00000001 | | |
| \$\$T1 | = 00000004 | | |
| \$\$T2 | = 00000004 | | |
| BUF | 0000008B | R | 03 |
| CHMRTN | 000003BE | R | 05 |
| CHM_CONT | 00000059 | R | 03 |
| CS1 | 00000031 | R | 02 |
| CS2 | 00000063 | R | 02 |
| CS3 | 00000090 | R | 02 |
| CS5 | 000000C3 | R | 02 |
| CTL\$GL PHD | ***** | X | 05 |
| CURRENT_TC | 00000004 | R | 03 |
| ERROR | = 00000002 | | |
| EXP | 000000D7 | R | 02 |
| GETBUF | 000000E3 | R | 03 |
| GET_LIST | 0000018B | R | 03 |
| INFO | = 00000003 | | |
| JPI\$ PPGCNT | = 0000030D | | |
| LIB\$SIGNAL | ***** | X | 05 |
| LOCK_AREA | 000001AB | R | 03 |
| MESSAGEL | 0000016F | R | 03 |
| ML | 000000DB | R | 03 |
| MODE | 00000069 | R | 03 |
| MODE_ID | 00000375 | R | 05 |
| MOD_MSG_CODE | 00000044 | R | 03 |
| MOD_MSG_PRINT | 000003A2 | R | 05 |
| MSGC | 00000083 | R | 03 |
| MSGVEC | 000000F1 | R | 02 |
| MSGVEC1 | 0000017B | R | 03 |
| PHD\$Q PRIVMSK | = 00000000 | | |
| PPG_CNT | 0000019B | R | 03 |
| PPG_CNT1 | 0000019F | R | 03 |
| PRINT_FAIL | 000002B1 | R | 05 |
| PRIVMSK | 00000051 | R | 03 |
| PRIV_ARGS | = 00000002 | | |
| PRV\$V PSWAPM | = 0000000C | | |
| PRVPRT | 00000050 | R | 03 |
| PURG | 000001B3 | R | 03 |
| PURGE_AREA | 000001A3 | R | 03 |
| PURGWS | 00000101 | R | 02 |
| PURGWS\$ INADR | = 00000004 | | |
| PURGWS\$ NARGS | = 00000001 | | |
| REG | 0000006D | R | 03 |
| REGNUM | 0000007F | R | 03 |
| REG_CHECK | 0000026F | R | 05 |
| REG_SAVE | 00000265 | R | 05 |
| REG_SAVE_AREA | 00000008 | R | 03 |
| RETADR | 0000005D | R | 03 |
| SATSSS80 | 00000000 | RG | 05 |
| SERV_NAME | 00000177 | R | 03 |
| SEVERE | = 00000004 | | |
| SHR\$K SHRDEF | = 00000001 | | |
| SHR\$ TEXT | = 00001130 | | |
| SS\$ NORMAL | = 00000001 | | |
| STATUS | 00000065 | R | 03 |
| STEP | = 00000003 | | |
| STPO | 0000003D | R | 05 |

| | | | |
|-----------------|------------|----|----|
| STP1 | 000000D2 | R | 05 |
| STP2 | 0000012F | R | 05 |
| STP3 | 000001A5 | R | 05 |
| ST\$V INHIB_MSG | = 0000001C | | |
| SUCCESS | = 00000001 | | |
| SY\$CMKRNL | ***** | GX | 05 |
| SY\$EXIT | ***** | GX | 05 |
| SY\$FAO | ***** | X | 05 |
| SY\$GETJPI | ***** | GX | 05 |
| SY\$HIBER | ***** | GX | 05 |
| SY\$LCKPAG | ***** | GX | 05 |
| SY\$PURGWS | ***** | GX | 05 |
| SY\$PUTMSG | ***** | GX | 05 |
| SY\$SETPRN | ***** | GX | 05 |
| SY\$SETPRV | ***** | GX | 05 |
| SY\$WAKE | ***** | GX | 05 |
| TEST_END | 000003C6 | R | 05 |
| TEST_MOD_BEGIN | 00000019 | R | 02 |
| TEST_MOD_FAIL | 0000002A | R | 02 |
| TEST_MOD_NAME | 00000000 | R | 02 |
| TEST_MOD_NAME_D | 00000009 | R | 02 |
| TEST_MOD_SUCC | 0000001F | R | 02 |
| TMD_ADDR | 0000004C | R | 03 |
| TMN_ADDR | 00000048 | R | 03 |
| TOUCH_PAGE | 00000000 | R | 04 |
| TPID | 00000000 | R | 03 |
| UETPS SATSMS | = 007480D9 | | |
| UETPS TEXT | = 00741133 | | |
| UM | 000000E5 | R | 02 |
| WARNING | = 00000000 | | |
| WS_STR | 00000108 | R | 02 |

+-----+
! Psect synopsis !
+-----+

| PSECT name | Allocation | PSECT No. | Attributes |
|------------|-------------------|-----------|---|
| . ABS . | 00000000 (0.) | 00 (0.) | NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE |
| \$AB\$\$ | 00000000 (0.) | 01 (1.) | NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE |
| RODATA | 0000011B (283.) | 02 (2.) | NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC PAGE |
| RWDATA | 000001BB (443.) | 03 (3.) | NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC PAGE |
| TOUCH PAGE | 00000600 (1536.) | 04 (4.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC PAGE |
| SATSSS80 | 000003C6 (966.) | 05 (5.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC PAGE |

+-----+
! Performance indicators !
+-----+

| Phase | Page faults | CPU Time | Elapsed Time |
|------------------------|-------------|-------------|--------------|
| Initialization | 29 | 00:00:00.11 | 00:00:00.68 |
| Command processing | 107 | 00:00:00.73 | 00:00:04.45 |
| Pass 1 | 374 | 00:00:12.01 | 00:00:30.06 |
| Symbol table sort | 0 | 00:00:01.53 | 00:00:02.77 |
| Pass 2 | 115 | 00:00:02.52 | 00:00:05.43 |
| Symbol table output | 11 | 00:00:00.09 | 00:00:01.03 |
| Psect synopsis output | 2 | 00:00:00.04 | 00:00:00.06 |
| Cross-reference output | 0 | 00:00:00.00 | 00:00:00.00 |
| Assembler run totals | 640 | 00:00:17.03 | 00:00:44.48 |

The working set limit was 1350 pages.

67614 bytes (133 pages) of virtual memory were used to buffer the intermediate code.

There were 50 pages of symbol table space allocated to hold 979 non-local and 12 local symbols.

495 source lines were read in Pass 1, producing 26 object records in Pass 2.

47 pages of virtual memory were used to define 42 macros.

+-----+
! Macro library statistics !
+-----+

| Macro library name | Macros defined |
|-------------------------------------|----------------|
| -\$255\$DUA28:[SYSLIB]STARLET.MLB;2 | 26 |
| -\$255\$DUA28:[SHRLIB]UETP.MLB;1 | 12 |
| -\$255\$DUA28:[SYS.OBJ]LIB.MLB;1 | 1 |
| -\$255\$DUA28:[SYSLIB]STARLET.MLB;2 | 0 |
| TOTALS (all libraries) | 39 |

1267 GETS were required to define 39 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:SATSSS80/OBJ=OBJ\$:SATSSS80 MSRC\$:SATSSS80/UPDATE=(ENH\$:SATSSS80)+EXECML\$/LIB+SHRLIB\$:UETP/LIB

0425 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY